In [40]:

```
import pandas as pd
import os

path = r"D:\kriti"

all_data = pd.DataFrame()

for file in os.listdir(path):
        df = pd.read_csv(os.path.join(path, file))
        all_data = pd.concat([all_data, df])

print(all_data.shape)
all_data.head()

#reading data from files as well as combining all files october,november and december sa
```

```
(63157, 6)
```

Out[40]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	295665	Macbook Pro Laptop	1	1700	12/30/19 00:01	136 Church St, New York City, NY 10001
1	295666	LG Washing Machine	1	600.0	12/29/19 07:03	562 2nd St, New York City, NY 10001
2	295667	USB-C Charging Cable	1	11.95	12/12/19 18:21	277 Main St, New York City, NY 10001
3	295668	27in FHD Monitor	1	149.99	12/22/19 15:13	410 6th St, San Francisco, CA 94016
4	295669	USB-C Charging Cable	1	11.95	12/18/19 12:38	43 Hill St, Atlanta, GA 30301

In [ ]:

```

```

In [9]:

```
all_data = all_data.dropna()
#dropping missing rows
```

In [11]:

```
all_data['Order Date'] = pd.to_datetime(all_data['Order Date'], errors='coerce')
all_data = all_data.dropna(subset=['Order Date'])
#covert date into real date format and remove wrong date format.
```

In [12]:

```
all_data['Quantity Ordered'] = pd.to_numeric(all_data['Quantity Ordered'])
all_data['Price Each'] = pd.to_numeric(all_data['Price Each'])
#covert quantity ordered n price into numeric format , so that calculations could be pos
```

In [13]:

```
all_data['Sales'] = all_data['Quantity Ordered'] * all_data['Price Each']
#calculating total sales
```

In [14]:

```
all_data['Month'] = all_data['Order Date'].dt.month
#extracting month from dates
```

In [17]:

```
all_data.groupby('Month')['Sales'].sum()
#calculating sales monthwise
```

Out[17]:

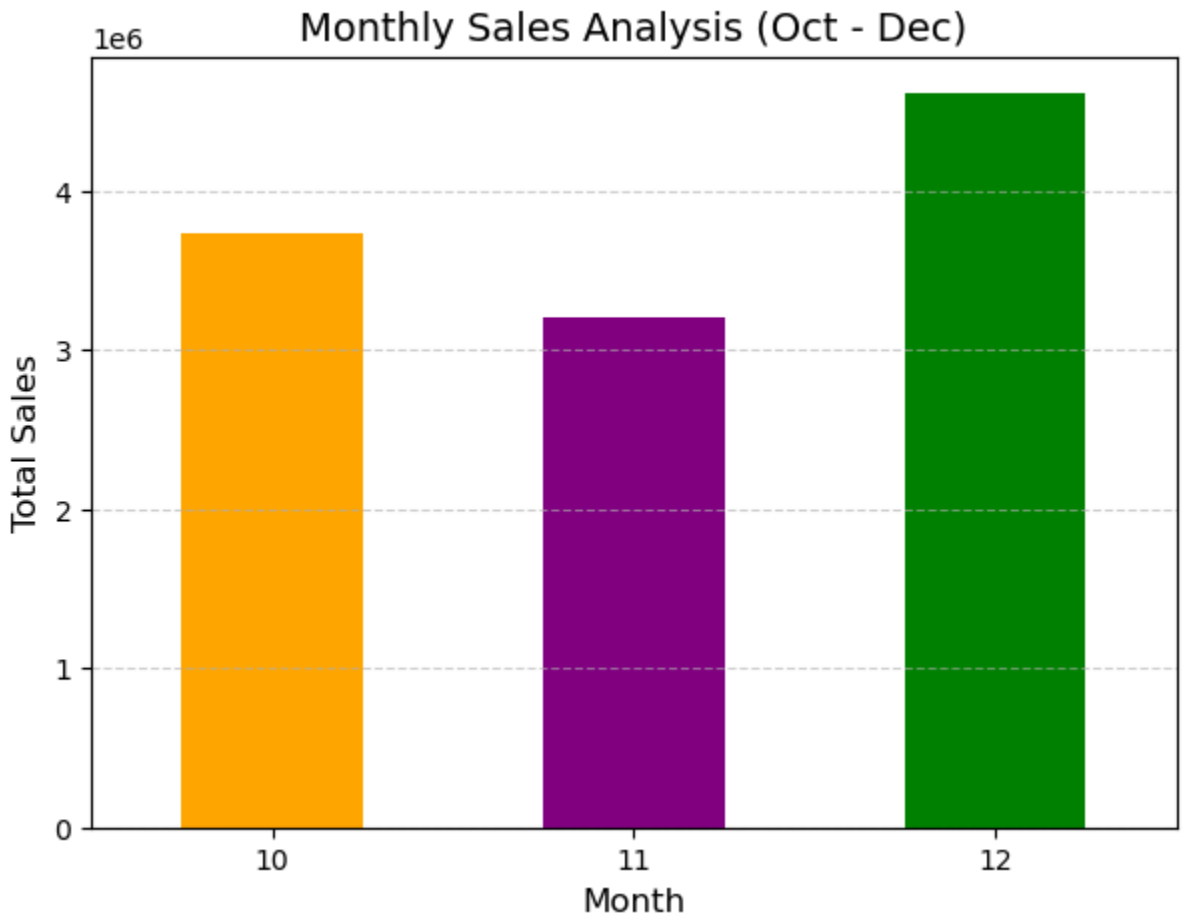
```
Month
10    3733373.57
11    3199603.20
12    4613443.34
Name: Sales, dtype: float64
```

In [16]:

```
all_data = all_data[all_data['Month'].isin([10, 11, 12])]
#calculating only for oct,nov.dec
```

In [38]:

```
import matplotlib.pyplot as plt

monthly_sales = all_data.groupby('Month')['Sales'].sum()

plt.figure(figsize=(7,5))
monthly_sales.plot(
    kind='bar',
    color=['orange', 'purple', 'green']
)

plt.title("Monthly Sales Analysis (Oct - Dec)", fontsize=14)
plt.xlabel("Month", fontsize=12)
plt.ylabel("Total Sales", fontsize=12)
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.6)

plt.show()

#sales analysis graph
```



In [21]:

```
top_products = all_data.groupby('Product')['Sales'].sum().sort_values(ascending=False)

top_products.head(5)
#calculating top products
```

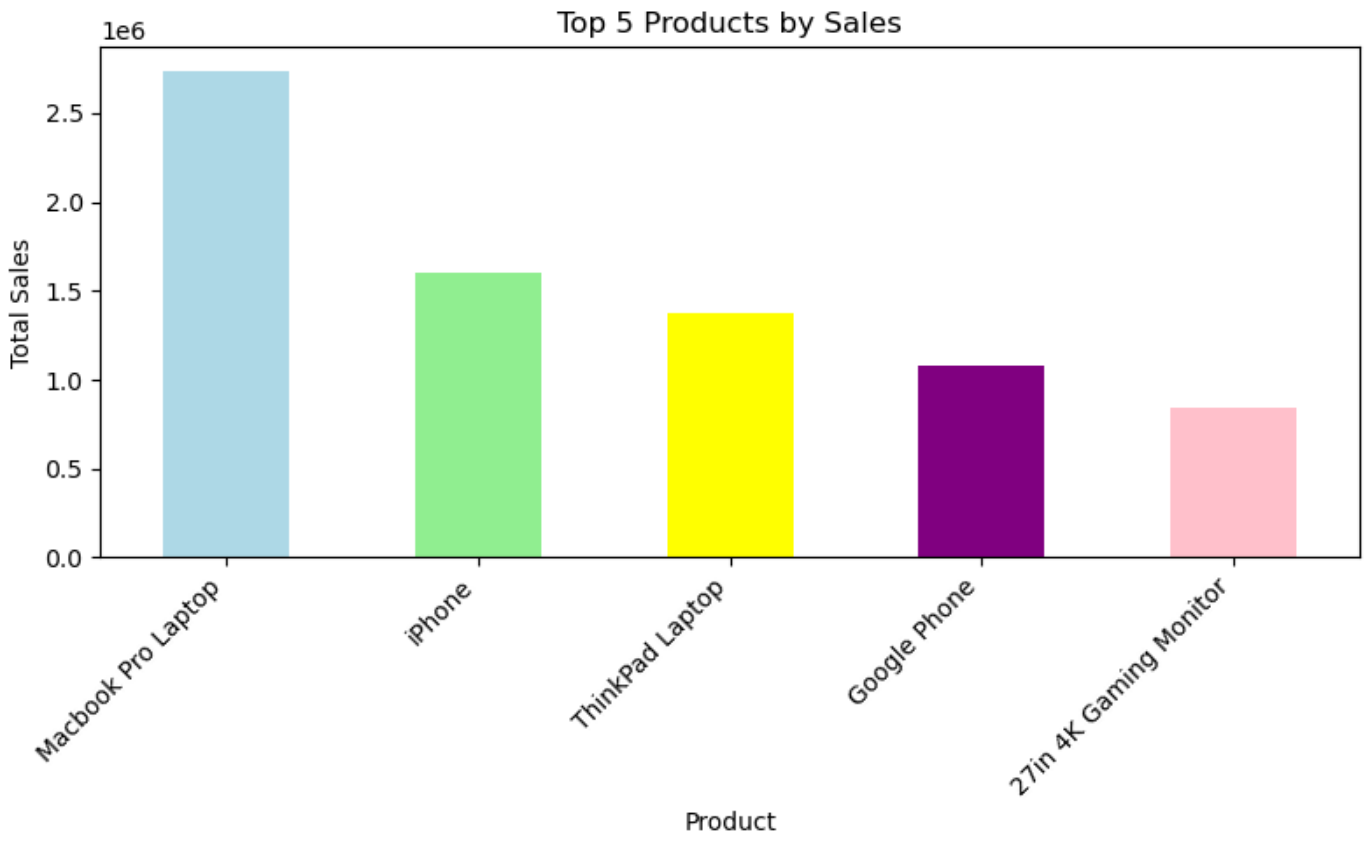
Out[21]:

```
Product
Macbook Pro Laptop        2733600.00
iPhone                    1600200.00
ThinkPad Laptop           1372986.27
Google Phone              1082400.00
27in 4K Gaming Monitor     842768.39
Name: Sales, dtype: float64
```

In [36]:

```
import matplotlib.pyplot as plt

top_products.head(5).plot(
    kind='bar',
    color=['lightblue', 'lightgreen', 'yellow', 'purple', 'pink'],
    figsize=(8,5)
)

plt.title("Top 5 Products by Sales")
plt.xlabel("Product")
plt.ylabel("Total Sales")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
```

```
plt.show()
#sales as per top 5 products
```



In [23]:

```
all_data['City'] = all_data['Purchase Address'].apply(lambda x: x.split(',')[1].strip())
```

In [24]:

```
city_sales = all_data.groupby('City')['Sales'].sum().sort_values(ascending=False)

city_sales.head(5)
#city wise sales
```

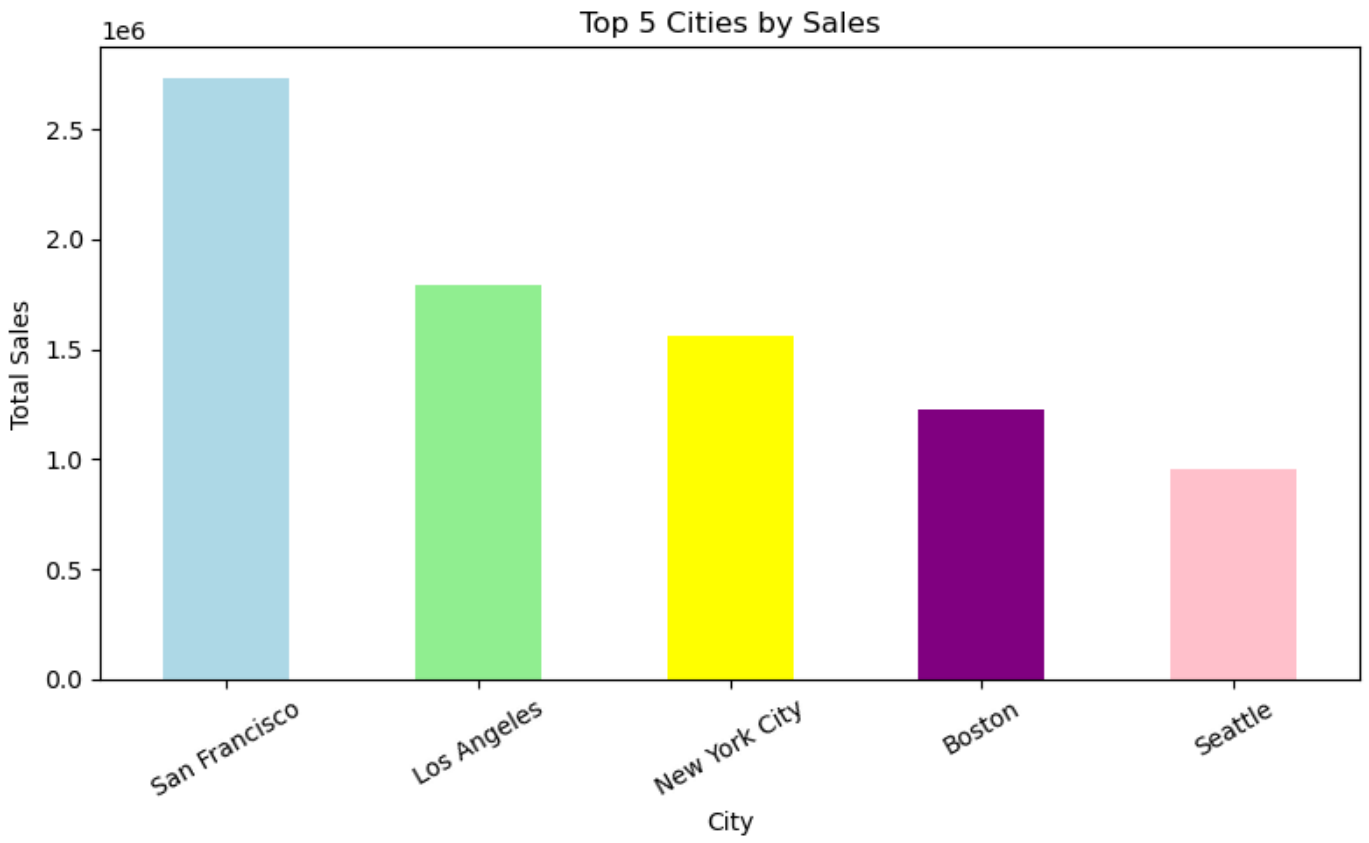
Out[24]:

```
City
San Francisco    2736542.70
Los Angeles      1795931.21
New York City    1561178.62
Boston           1228071.58
Seattle           956254.32
Name: Sales, dtype: float64
```

In [39]:

```
import matplotlib.pyplot as plt

city_sales.head(5).plot(
    kind='bar',
    figsize=(8,5),
    color=['lightblue', 'lightgreen', 'yellow', 'purple', 'pink']
)

plt.title("Top 5 Cities by Sales")
plt.xlabel("City")
plt.ylabel("Total Sales")
```

```
plt.xticks(rotation=30)
plt.tight_layout()
plt.show()
```



In [27]:

```
hourly_orders = all_data.groupby('Hour').count()['Order ID']
hourly_orders
#calculating sales per hour
```

Out[27]:

```
Hour
0     1292
1      802
2      395
3      282
4      313
5      435
6      866
7     1409
8     2020
9     2865
10    3734
11    4200
12    4309
13    4143
14    3701
15    3467
16    3527
17    3679
18    4148
19    4429
20    4111
21    3672
```

```
22    2904
23    2116
Name: Order ID, dtype: int64
```

In [28]:

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(8,5))
plt.plot(hourly_orders.index, hourly_orders.values, marker='o')

plt.title("Orders by Hour of Day")
plt.xlabel("Hour")
plt.ylabel("Number of Orders")
plt.grid(True)
plt.show()
# showing through graph sales per hour
```

In [29]:

```python
def price_category(price):
    if price < 50:
        return "Low Price"
    elif price < 200:
        return "Medium Price"
    else:
        return "High Price"

all_data['Price Category'] = all_data['Price Each'].apply(price_category)
# analysing price low ,medium and high
```

In [30]:

```
all_data.groupby('Price Category')['Sales'].sum()
#every product a label
```

Out[30]:

```
Price Category
High Price      9411113.95
Low Price        366070.43
Medium Price    1769235.73
Name: Sales, dtype: float64
```

In [31]:

```
total_revenue = all_data['Sales'].sum()
total_orders = all_data['Order ID'].nunique()

average_order_value = total_revenue / total_orders
average_order_value
#average order value
```

Out[31]:

```
np.float64(191.5400967121197)
```

In [32]:

```
from itertools import combinations
from collections import Counter

grouped = all_data.groupby('Order ID')['Product'].apply(list)

count = Counter()

for products in grouped:
    count.update(Counter(combinations(products, 2)))

count.most_common(5)
#creating list of every product
```

Out[32]:

```
[(('Google Phone', 'USB-C Charging Cable'), 308),
 (('iPhone', 'Lightning Charging Cable'), 301),
 (('iPhone', 'Wired Headphones'), 159),
 (('Google Phone', 'Wired Headphones'), 152),
 (('iPhone', 'Apple Airpods Headphones'), 130)]
```

In [ ]:

```

```